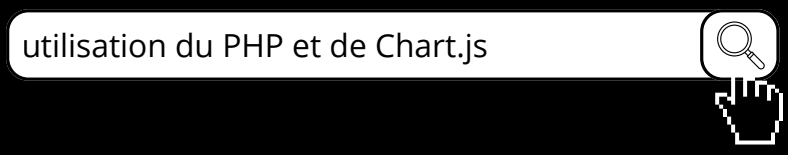
PROJET WEB

PERROT Loick



Le but de ce projet était de créer un site web qui interroge une base de données afin de créer des graphiques par rapport à des données issues de capteurs de températures dans des bâtiments de l'université.

Développement front-End

J'ai créé des fichiers HTML et CSS afin de structurer mon code et de styliser ma page pour la rendre plus conviviale.

```
<body onload="init()">
    <h1 id="titre_page">Tableau de bord énergétique</h1>

    <select id="selectbat" onchange="changeBat()">
    </select>

    <h1>Batiment : <span id="bat">Pas de sélection</span></h1>

    <div class="table-scroll">
        <table>
            <thead>
                <tr>
                    <th>Batiment</th>
                    <th>Capteur</th>
                    <th>Dernière mesure</th>
                </tr>
            </thead>
            <tbody></tbody>
        </table>
    </div>

    <hr>

    <div style="margin-bottom: 15px;">
        <label for="selectCapteurGraph"><strong>Voir le graphique du capteur : </strong></label>
        <select id="selectCapteurGraph" onchange="changeCapteurGraph()">
            <option value="">-- Sélectionnez un capteur --</option>
        </select>
    </div>

    <div class="chart-card">
        <canvas id="graphMesure"></canvas>
    </div>

    <div class="chart-card-type">
        <div id="graphtype"></div>
    </div>
</body>
</html>
```

Figure 1 extrait du code HTML

```css
@import url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;600;800&display=swap');

/* VARIABLES (Couleurs et Ombres)*/
:root {
    --primary-gradient: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
    --hover-gradient: linear-gradient(135deg, #764ba2 0%, #667eea 100%);
    --bg-overlay-start: rgba(240, 242, 245, 0.85);
    --bg-overlay-end: rgba(240, 242, 245, 0.95);
    --card-bg: #ffffff;
    --text-color: #2d3748;
    --shadow-soft: 0 10px 15px -3px rgba(0, 0, 0, 0.1), 0 4px 6px -2px rgba(0, 0, 0, 0.05);
    --shadow-hover: 0 20px 25px -5px rgba(0, 0, 0, 0.1), 0 10px 10px -5px rgba(0, 0, 0, 0.04);
}

/*GLOBAL & BACKGROUND*/
body {
    font-family: 'Poppins', sans-serif;

    /* Image de fond avec voile semi-transparent */
    background:
        linear-gradient(var(--bg-overlay-start), var(--bg-overlay-end)),
        url('https://images.unsplash.com/photo-1509391366360-2e959784a276?q=80&w=1920&auto=format&fit=crop');

    background-size: cover;
    background-position: center;
    background-attachment: fixed;
    background-repeat: no-repeat;

    color: var(--text-color);
    margin: 0;
    padding: 40px 20px;
    display: flex;
    flex-direction: column;
    align-items: center;
    min-height: 100vh;
}
```

Figure 2 extrait du code CSS

Développement back end

La majeure partie du travail réside dans cette section. J'ai structuré mes fichiers PHP en dossiers et sous-dossiers pour garantir un code propre et organisé. Le répertoire racine, nommé **api**, contient plusieurs sous-dossiers regroupant les fonctions chargées d'appeler les requêtes SQL. Ces dernières sont définies dans le dossier **objects**, qui inclut également un répertoire **database** dédié à la connexion à la base de données.

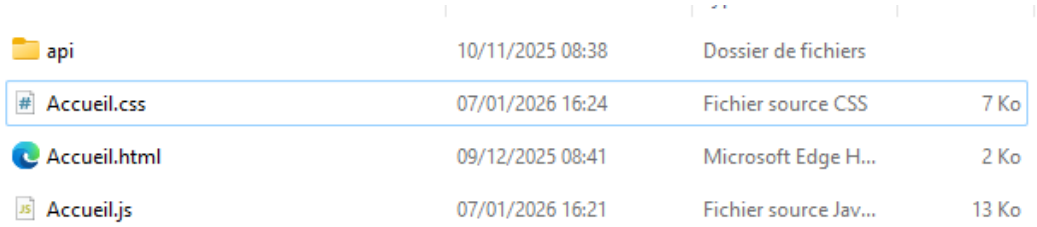


Figure 3 Agencement des fichiers du code

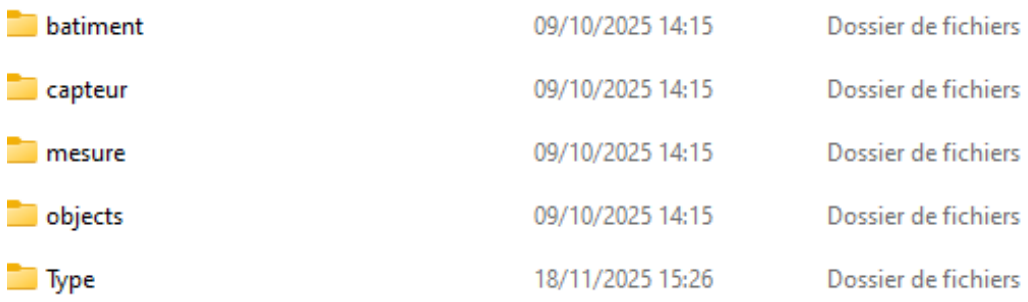


Figure 4 Agencement des dossiers dans le dossier api

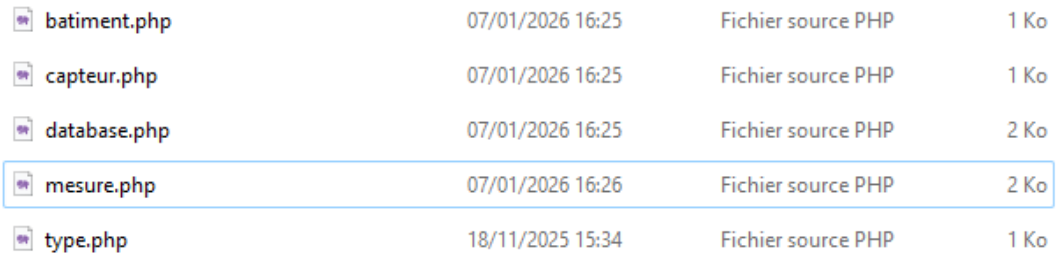


Figure 5 Agencement des fichiers du code dans le dossier objects

```php
<?php
//inclusion du contenu du fichier database -> récupération de la class Database
require_once __DIR__."/database.php";

class Batiment{

    function getAll(){
        //définition de la requête
        $query = "SELECT id, nom FROM Batiment ORDER BY nom ASC;";
        //on définit les paramètres (si nécessaire)
        //exécution et récupération des données
        return Database::execute($query);
    }
}
```

Figure 6 Exemple d'une fonction PHP du dossier objects

```php
<?php
require_once __DIR__."/../objects/batiment.php";

// required headers
//header("Access-Control-Allow-Headers: Content-Type"); //pour spécifier le format des données que l'on accepte, utile pour POST et PUT
header("Access-Control-Allow-Methods: GET"); //pour n'autoriser que les requêtes GET
header("Content-Type: application/json; charset=UTF-8"); //Pour spécifier le format du retour : JSON avec encodage UTF-8

$batimentObj = new Batiment();
echo json_encode($batimentObj->getAll());
```

Figure 7 Exemple avec la fonction du dossier bâtiment appelant la fonction précédente

Une fois les données transmises par le PHP, elles sont récupérées en JavaScript via l'API **Fetch**, puis traitées par les fonctions correspondantes pour dynamiser l'interface.

```javascript
//Charge la dernière mesure pour un capteur spécifique
async function loadDetailsMesure(idCapteur, td) {
    try {
        const response = await fetch("api/mesure/get-mesure.php?idcapt=" + idCapteur);// lancement
        if (!response.ok) throw new Error(`Erreur HTTP: ${response.status}`);//message d'erreur

        const details = await response.json();
        afficheDetailsMesure(details, td);//appel de la fonction qui affiche les données
    } catch (error) {
        console.error("Erreur loadDetailsMesure:", error);
        td.innerHTML = "Erreur";
    }
}
```

Figure 8 Exemple du fetch pour récupérer les données du PHP.

Après avoir généré l'ensemble des données, nous les intégrons à l'interface sous forme de tableaux ou de graphiques en utilisant les fonctions adéquates.

```
//   GESTION DU GRAPHIQUE LINEAIRE
function initGraphMesures(){
    const canvas = document.querySelector("#graphMesure");
    graphMesure = new Chart(canvas, {
        type: "line",
        data: { labels: [], datasets: [] },
        options: {
            responsive: true,
            plugins: {
                legend: { position: 'top' },
                title: { display: true, text: "Historique des mesures" }
            }
        }
    });
}
```

Figure 9 Exemple de fonction de création de graphique

Une fois l'ensemble des fonctions développé et le style appliqué au fichier HTML, nous obtenons un site fonctionnel, interactif et capable d'interroger la base de données en temps réel.

Figure 10 Tableau contenant les capteurs et leurs mesures (pour l'exemple nous avions un seul capteur avec des mesures)

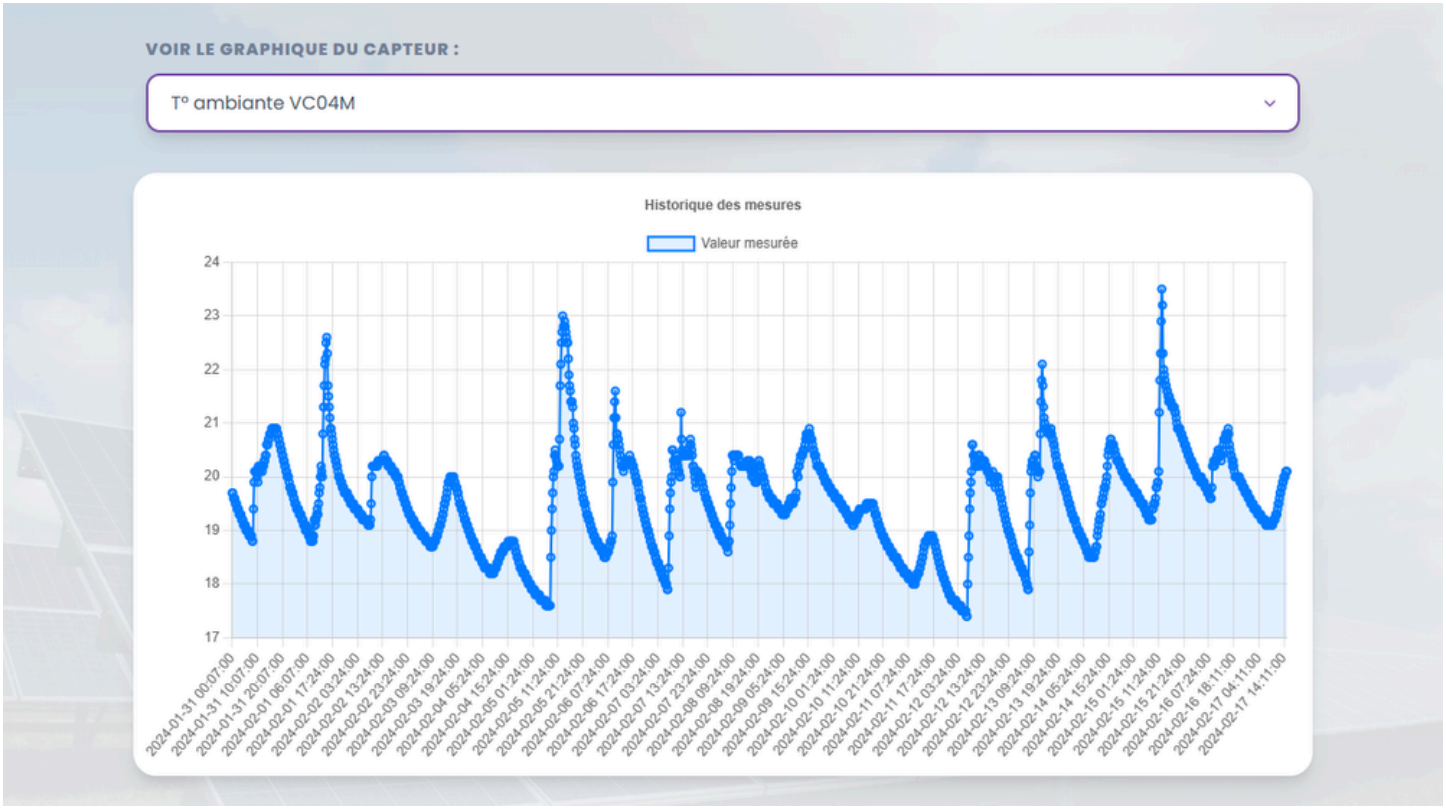


Figure 11 Graphique des températures pour un capteur sélectionné